A REPORT ON LANGUAGE IDENTIFICATION IN SPEECH USING ARTIFICIAL NEURAL NETWORK

July 12, 2018

Supervised by: Dr. Vrijendra Singh
HoD, Information Technology
IIIT-Allahabad

Anand Tiwari
Summer Intern, IIIT-Allahabad
Department of Information Technology

Contents

1 INTRODUCTION

Automatic Speech Language Identification (LID) is the problem to accurately identify the language from a small speech sample.Humans are the best known LIDs in the world today but it will not be strange if their supremacy is taken away by machines in the near future.

The quest for LIDs started way back in 1970s but real development in this field has taken place in this century.In the present era of Artificial Intelligence (AI) Revolution, LID systems are largely in demand finding their applications in speech to speech translation,front-end for automatic speech recognition and front-end for human operators.LID systems are also very useful in multilingual public service windows.

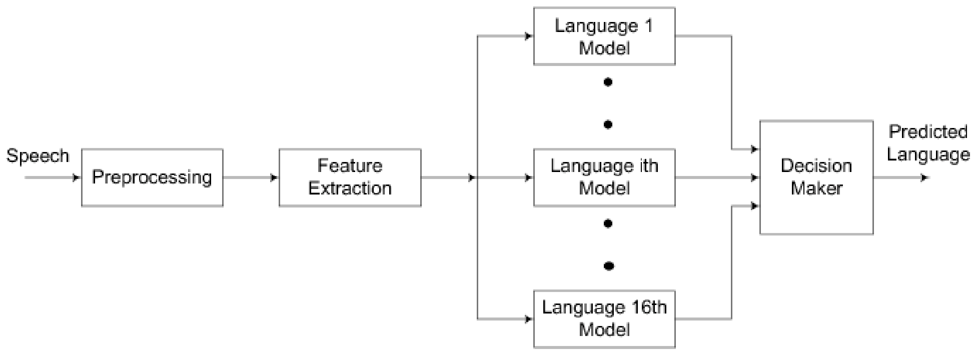


Figure 1: Language Identification System

Language Identification systems can be basically classified into two categories : Explicit and Implicit.Explicit LIDs uses speech recognition concepts,i.e., these systems first get phonemes out of the speech sample and then classify it for multiple languages.These systems require segmented and labeled speech corpus.In Implicit LIDs, language is identified by raw speech data only.There is no need to get phoneme sequences out of speech.Acoustic features of language are enough for identification.

In this paper,I have worked on LID system of four Indian languages,namely Indian English,Hindi,Bangla and Telugu.Since very few works have been carried out for Indian languages, there is scarcity of speech corpus.I have build my own speech corpus for this task.Using MFCCs as the acoustic features,I have modeled LIDs using simple Artificial Neural Network(ANN).I have also implemented a LID using Gaussian Mixture Models(GMMs).

This paper not only uses MFCCs for identification purpose,but also explores it with Shifted Delta Cepstrum(SDC) features to get improved results.

Keywords: Language Identification,Speech Processing,Signal Processing,Voice Activity Detection,Speech Science ,Speech Signal Processing

2 Building Speech Database

Unavailability of proper speech corpus forced us to build our own small speech corpus.Database covers speech samples in 4 Indian languages, namely Hindi,Indian English,Bangla and Telugu.Database is recorded by various speakers speaking different languages to make our data open to variation in accents, variation in dialects etc.

Speech samples each of 5 to 10 seconds is recorded. A speaker is used for more than one samples and also for more than one language,i.e.,North Indians recorded more than 2 samples each for Hindi and Indian English.

Details related to speech samples :

- Format - Waveform Audio File Format (.wav)
- Sampling Rate - 16000 Hz
- Noise - Considerable Noise (Required to be removed)

Speech samples collected for each language:

- Training Corpus
  1. Bangla - 15
  2. Indian English - 23
  3. Hindi - 20
  4. Telugu - 12
- Testing Corpus
  1. Bangla - 2
  2. Indian English - 4
  3. Hindi - 2
  4. Telugu - 2

3 Data Preprocessing

Simple speech generally carries a lot of noise while passing through a communication channel.Thus, it is necessary to preprocess our speech corpus to remove noise from the samples.It is preferable to make our speech corpus noise free for better language identification.

Voice Activity Detection is one such technique in which presence of human speech is detected.It can avoid unnecessary coding/transmission of silence packets in Voice over Internet Protocol applications, saving on computation and on network bandwidth.

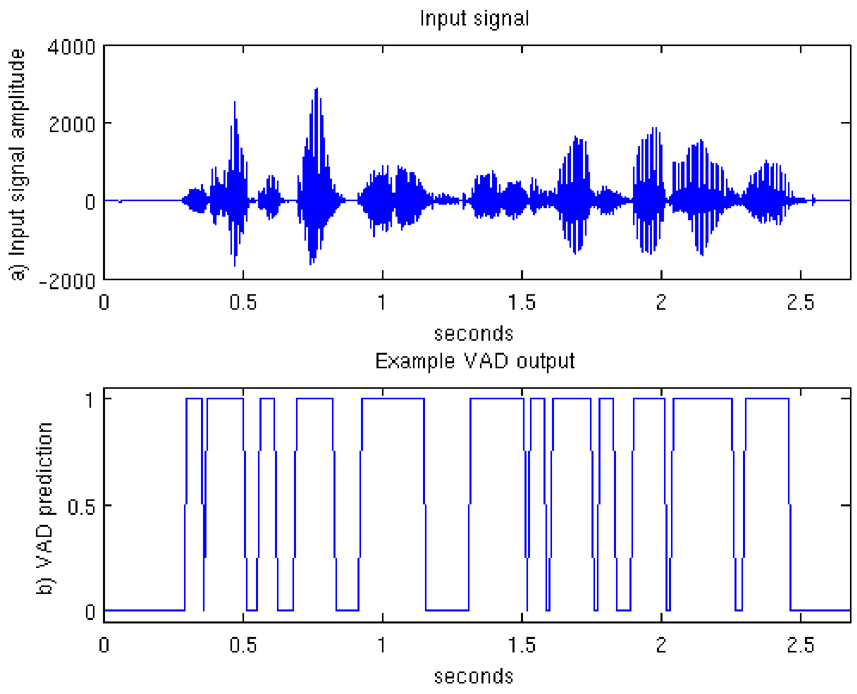
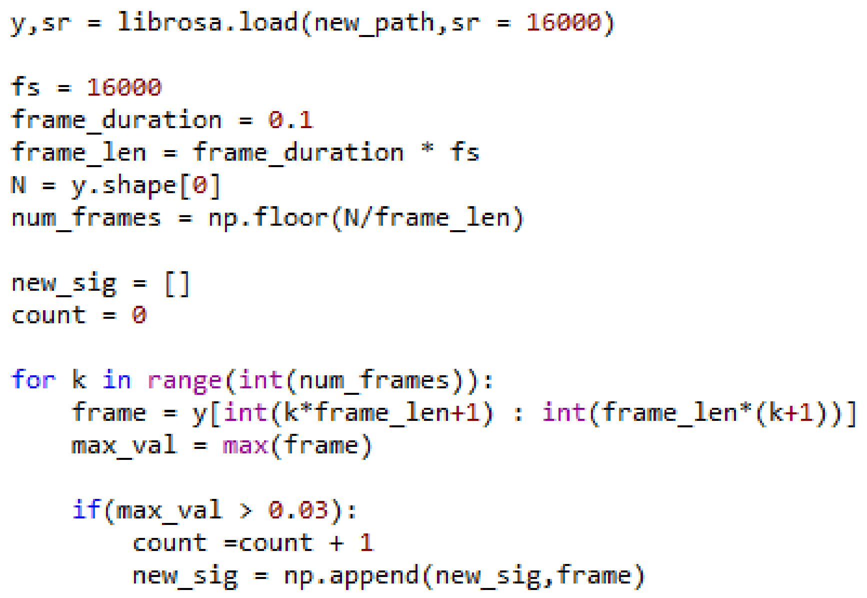


Figure 2: Voice Activity Detection

In this work, I have used the most basic approach to remove noise out of speech samples.Considering our recording setup and surroundings, we visualized noise in the channel to be of amplitude around 0.03.Thus, we modeled our samples to remove frames with maximum amplitude of 0.03.

At first, we break the speech signal into set of frames with the sampling rate of 16000 Hz.I have used python LIBROSA library to divide speech signal into frames.Then for each frame, we check whether the maximum amplitude for the given frame is more than 0.03.If the condition does not satisfies, then that frame is removed from the speech signal.



```python
y,sr = librosa.load(new_path,sr = 16000)

fs = 16000
frame_duration = 0.1
frame_len = frame_duration * fs
N = y.shape[0]
num_frames = np.floor(N/frame_len)

new_sig = []
count = 0

for k in range(int(num_frames)):
    frame = y[int(k*frame_len+1) : int(frame_len*(k+1))]
    max_val = max(frame)

    if(max_val > 0.03):
        count =count + 1
        new_sig = np.append(new_sig,frame)
```

Figure 3: VAD Code

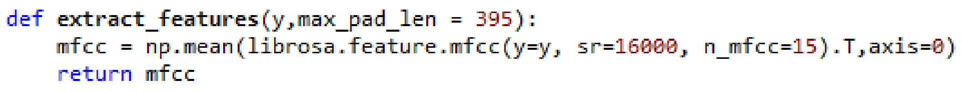
This ensures that there is no frame in speech signal with only noise in it.Thus,VAD will improve the performance of the LID system in terms of accuracy and time.VAD is very helpful in noisy environments.

4 Feature Extraction

Feature Extraction is one of the most important part of a LID system. It is required to choose acoustic feature wisely considering the problem to be solved.There are a number of acoustic features which can be used such as Chroma features, tempogram,Linear Prediction Cepstral Coefficients (LPCCs) etc. I have used the most common acoustic feature used in Automatic Recognition Systems(ASR), the Mel Frequency Cepstral Coefficients(MFCCs) for this purpose.

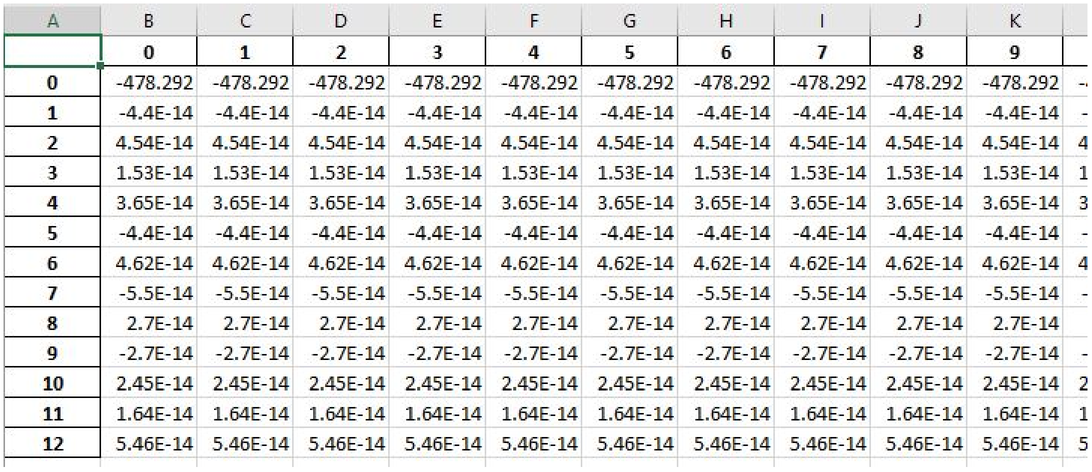
4.1 Mel Frequency Cepstral Coefficients (MFCCs)

MFCCs are the most preferred choice of spectral features in ASR systems.These are also used in music annotation and various information retrieval.I have generated 15-dimensional MFCC features for each frame of the speech sample.Since we can not feed MFCC for each frame in our neural model, we take the mean of all the MFCCs. This provides us the rough estimate of acoustic feature of the given speech sample belonging to a particular language.



```
def extract_features(y,max_pad_len = 395):
    mfcc = np.mean(librosa.feature.mfcc(y=y, sr=16000, n_mfcc=15).T,axis=0)
    return mfcc
```

Figure 4: Mel Frequency Cepstral Coefficients



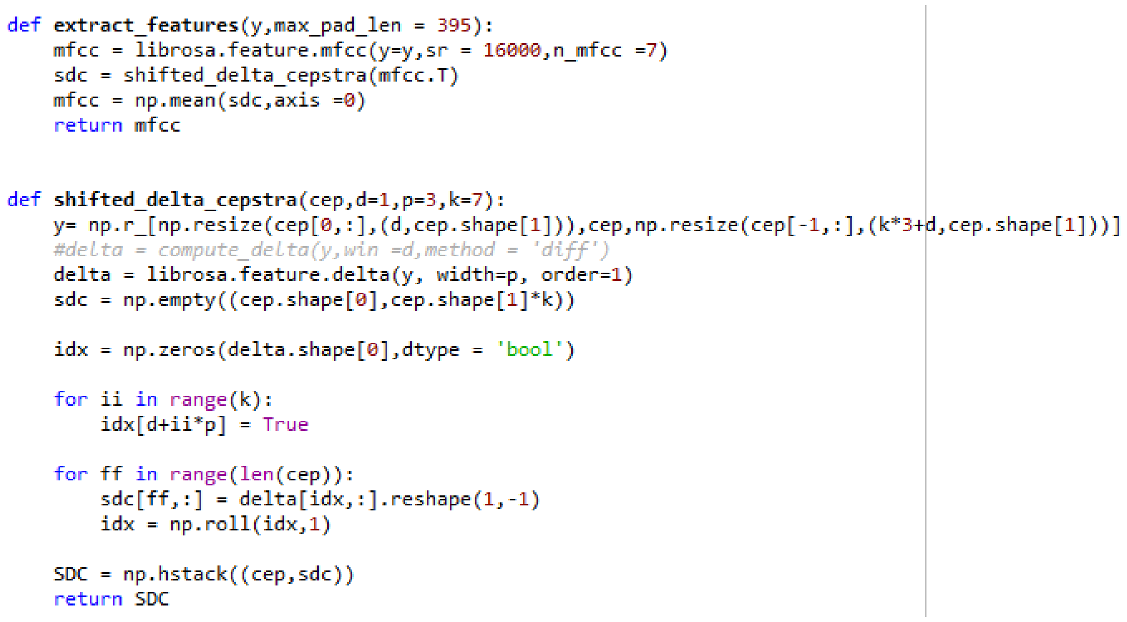
A	B	C	D	E	F	G	H	I	J	K	
	0	1	2	3	4	5	6	7	8	9	
0	-478.292	-478.292	-478.292	-478.292	-478.292	-478.292	-478.292	-478.292	-478.292	-478.292	-
1	-4.4E-14	-4.4E-14	-4.4E-14	-4.4E-14	-4.4E-14	-4.4E-14	-4.4E-14	-4.4E-14	-4.4E-14	-4.4E-14	-
2	4.54E-14	4.54E-14	4.54E-14	4.54E-14	4.54E-14	4.54E-14	4.54E-14	4.54E-14	4.54E-14	4.54E-14	4
3	1.53E-14	1.53E-14	1.53E-14	1.53E-14	1.53E-14	1.53E-14	1.53E-14	1.53E-14	1.53E-14	1.53E-14	1
4	3.65E-14	3.65E-14	3.65E-14	3.65E-14	3.65E-14	3.65E-14	3.65E-14	3.65E-14	3.65E-14	3.65E-14	3
5	-4.4E-14	-4.4E-14	-4.4E-14	-4.4E-14	-4.4E-14	-4.4E-14	-4.4E-14	-4.4E-14	-4.4E-14	-4.4E-14	-
6	4.62E-14	4.62E-14	4.62E-14	4.62E-14	4.62E-14	4.62E-14	4.62E-14	4.62E-14	4.62E-14	4.62E-14	4
7	-5.5E-14	-5.5E-14	-5.5E-14	-5.5E-14	-5.5E-14	-5.5E-14	-5.5E-14	-5.5E-14	-5.5E-14	-5.5E-14	-
8	2.7E-14	2.7E-14	2.7E-14	2.7E-14	2.7E-14	2.7E-14	2.7E-14	2.7E-14	2.7E-14	2.7E-14	
9	-2.7E-14	-2.7E-14	-2.7E-14	-2.7E-14	-2.7E-14	-2.7E-14	-2.7E-14	-2.7E-14	-2.7E-14	-2.7E-14	-
10	2.45E-14	2.45E-14	2.45E-14	2.45E-14	2.45E-14	2.45E-14	2.45E-14	2.45E-14	2.45E-14	2.45E-14	2
11	1.64E-14	1.64E-14	1.64E-14	1.64E-14	1.64E-14	1.64E-14	1.64E-14	1.64E-14	1.64E-14	1.64E-14	1
12	5.46E-14	5.46E-14	5.46E-14	5.46E-14	5.46E-14	5.46E-14	5.46E-14	5.46E-14	5.46E-14	5.46E-14	5

Figure 5: 13 dimensional MFCCs

4.2 Shifted Delta Cepstral (SDC) Features

Shifted Delta Cepstral (SDC) Features are used for incorporating temporal information about speech signal into the feature vectors or MFCCs.SDC features are created by stacking delta cepstra,computed along multiple such frames.SDCs are parametrized as:
N - number of cepstral coefficients computed at each time $(c_0,....,c_N - 1)$
d - time advance and delay for delta computation
k - number of blocks where delta coefficients are added to form final feature vector
p - time shift between consecutive blocks

I have used a 7-1-3-7 configuration SDC for our model.For our 7-dimensional MFCCs, it produces 56-dimensional SDC feature vectors.



```
def extract_features(y,max_pad_len = 395):
    mfcc = librosa.feature.mfcc(y=y,sr = 16000,n_mfcc =7)
    sdc = shifted_delta_cepstra(mfcc.T)
    mfcc = np.mean(sdc,axis =0)
    return mfcc


def shifted_delta_cepstra(cep,d=1,p=3,k=7):
    y= np.r_[np.resize(cep[0,:],(d,cep.shape[1])),cep,np.resize(cep[-1,:],(k*3+d,cep.shape[1]))]
    #delta = compute_delta(y,win =d,method = 'diff')
    delta = librosa.feature.delta(y, width=p, order=1)
    sdc = np.empty((cep.shape[0],cep.shape[1]*k))

    idx = np.zeros(delta.shape[0],dtype = 'bool')

    for ii in range(k):
        idx[d+ii*p] = True

    for ff in range(len(cep)):
        sdc[ff,:] = delta[idx,:].reshape(1,-1)
        idx = np.roll(idx,1)

    SDC = np.hstack((cep,sdc))
    return SDC
```

Figure 6: Shifted Delta Cepstrum

At first, we compute 7-dimensional MFCCs for each frame of speech sample.These MFCCs are then fed to get 56-dimensional SDC features for speech sample.At last, we take the mean of SDCs to get required SDC for that speech sample.

5 Neural Network Language Identification

This paper employs simple neural network for language identification task.Mel Frequency Cepstral Coefficients for each frame are fed as input to the neural network along with the concerned language which are fed as one-hot vectors.For K - language classification, the output layer produces K outputs,the log-likelihood for all K- languages.

5.1 Cost Function

I have used cross entropy cost function for our neural network model.Our purpose is to minimize the cost function during the training period.

$$J(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} H(p_n, q_n) = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)\right]$$

Figure 7: Cross entropy cost function

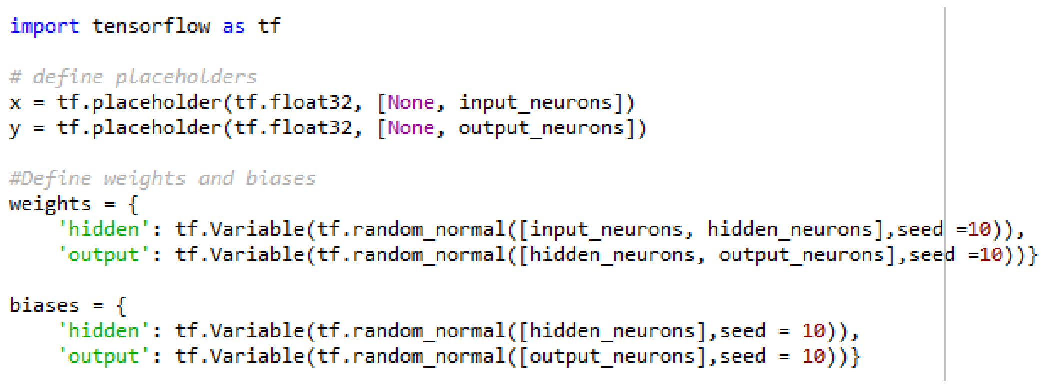
Here, N is the number of samples used for training,y_n is the target output and ˆy_n is the calculated output.

5.2 Neural Network Architecture

I have used simple neural network with one input layer,one output layer and one or more hidden layers.The number of neurons in each layer depends upon the acoustic feature used for training our model.Weights and Biases are randomly defined for each link between two neurons.At the base level, I have used one hidden layer.It is also tested with 2 and 3 hidden layers.

- **MFCCs**
  Input Neurons -15
  Hidden Neurons -10
  Output Neurons -4

- **MFCCs with SDC features**
  Input Neurons - 56
  Hidden Neurons - 20
  Output Neurons - 4



```
import tensorflow as tf

# define placeholders
x = tf.placeholder(tf.float32, [None, input_neurons])
y = tf.placeholder(tf.float32, [None, output_neurons])

#Define weights and biases
weights = {
    'hidden': tf.Variable(tf.random_normal([input_neurons, hidden_neurons],seed =10)),
    'output': tf.Variable(tf.random_normal([hidden_neurons, output_neurons],seed =10))}

biases = {
    'hidden': tf.Variable(tf.random_normal([hidden_neurons],seed = 10)),
    'output': tf.Variable(tf.random_normal([output_neurons],seed = 10))}
```

Figure 8: Initialize parameters

5.3 Model Training

Our model is trained through forward back propagation technique.I have used sigmoid activation function for this task,i.e.,if **z** is the input to any layer,then it produces **a** as the output,where

$$a = g(z) = \frac{1}{1+e^{-z}}$$

The output of jth layer,a_j is transformed as $z_{j+1} = w_{(j+1)j}a_j$to be fed as input to the (j+1)th layer. The output of the last layer is the output from forward propagation step of our neural structure.

Forward propagation:

$$\begin{aligned}
a^{(1)} &= x \\
z^{(2)} &= \Theta^{(1)}a^{(1)} \\
a^{(2)} &= g(z^{(2)}) \quad (\text{add } a_0^{(2)}) \\
z^{(3)} &= \Theta^{(2)}a^{(2)} \\
a^{(3)} &= g(z^{(3)}) \quad (\text{add } a_0^{(3)}) \\
z^{(4)} &= \Theta^{(3)}a^{(3)} \\
a^{(4)} &= h_\Theta(x) = g(z^{(4)})
\end{aligned}$$

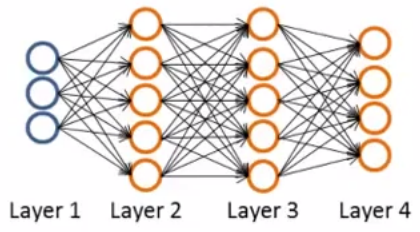


Figure 9: Forward Propagation

The output of the forward propagation step is compared with the actual target output and error is computed. This error is propagated backward and weights are adjusted in a way that it minimizes cross entropy cost function,**J(w)**. For optimization purpose,**Gradient Descent Optimizer** is used which calculates $\delta J(w)/\delta w_{ji}$ and minimizes the cost function.

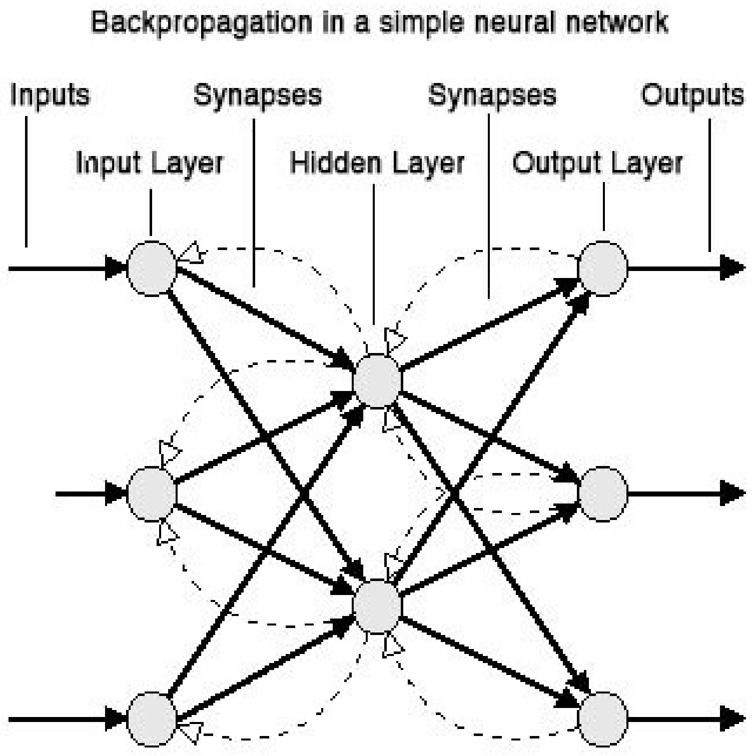
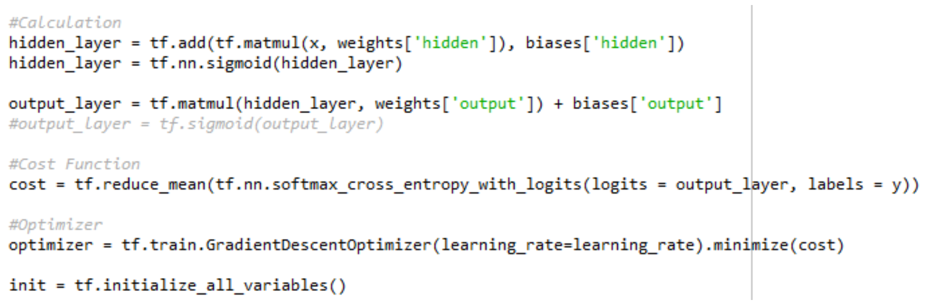


Figure 10: Backward Propagation

I have used **Tensorflow** python library for above explained task.This library provide pretty efficient algorithms for faster computation.



```
#Calculation
hidden_layer = tf.add(tf.matmul(x, weights['hidden']), biases['hidden'])
hidden_layer = tf.nn.sigmoid(hidden_layer)

output_layer = tf.matmul(hidden_layer, weights['output']) + biases['output']
#output_layer = tf.sigmoid(output_layer)

#Cost Function
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits = output_layer, labels = y))

#Optimizer
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate).minimize(cost)

init = tf.initialize_all_variables()
```

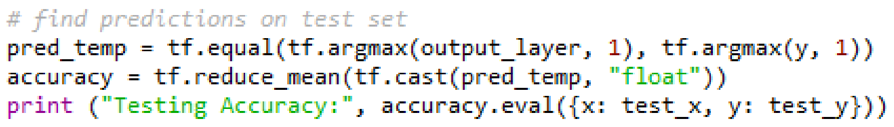
Figure 11: Forward Backward Propagation in ANN

Our model is trained for each of the speech samples a number of times in order to efficiently adjust the weights.These adjusted weights are then used for predictions for our testing data.The K-dimensional output prediction is actually the log-likelihood to indicate how likely it is to classify our input into one of the K languages. The predicted language K* is the maximum of all:

$$K^* = argmax_{k\in[1...K]}(h_\theta(x)_k)$$

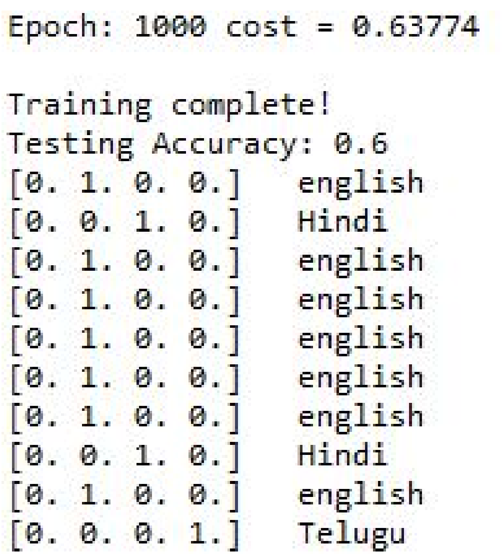
5.4 Performance Evaluation

After the model is trained,it is evaluated on out testing speech samples.Different testing parameters are set for proper predictions.I have tested our model for different set of parameters and proper evaluations have been made.



```
# find predictions on test set
pred_temp = tf.equal(tf.argmax(output_layer, 1), tf.argmax(y, 1))
accuracy = tf.reduce_mean(tf.cast(pred_temp, "float"))
print ("Testing Accuracy:", accuracy.eval({x: test_x, y: test_y}))
```

Figure 12: Model Testing



```
Epoch: 1000 cost = 0.63774

Training complete!
Testing Accuracy: 0.6
[0. 1. 0. 0.]   english
[0. 0. 1. 0.]   Hindi
[0. 1. 0. 0.]   english
[0. 1. 0. 0.]   english
[0. 1. 0. 0.]   english
[0. 1. 0. 0.]   english
[0. 1. 0. 0.]   english
[0. 0. 1. 0.]   Hindi
[0. 1. 0. 0.]   english
[0. 0. 0. 1.]   Telugu
```

Figure 13: Model Predictions

Different parameters on which model has been tested includes:

5.4.1 Learning Rate

Learning Rate of our Gradient Descent Optimizer is used for weights adjustment in our Neural Network model.I have tested our model for different learning rates while keeping **1000 epochs** for our model.

Table 1: Accuracy of model for different Learning rates

Learning Rate	Accuracy of Neural Network with MFCCs	Accuracy of Neural Network with MFCCs and SDC features
0.01	60	50
0.05	**60**	**80**
0.08	60	80
0.1	60	80
0.2	50	70

This shows that learning rate should be chosen carefully. Both very small learning rates and very high learning rates disturbs our model.

5.4.2 Number of epochs (iterations)

Same as learning rate,number of epochs should also be chosen with great accuracy.Validation set is used to get the optimal number of iterations.I tested model for different number of iterations keeping **learning rate fixed to 0.05**

Table 2: Model accuracy with different number of epochs

Number of Epochs	Accuracy of Neural Network with MFCCs	Accuracy of Neural Network with MFCCs and SDC features
1000	60	80
2000	60	80
5000	60	80
10000	60	80
15000	60	80

Though,there is no change in the accuracy of the model,it will show its impact for large testing data.We are performing our test on a small set of data which might not show the correct trends.

5.4.3 Number of hidden layers

Number of layers play a significant role in an ANN.Number of layers and their neurons should be chosen with care.I have tested my model for more than 1 hidden layers for different configurations of hidden neurons.

- **Two Hidden Layer Neural Network**
  15-dimensional input features are processed to give 4-dimensional output vector.

Table 3: Model accuracy for Neural Network with MFCC features

Neurons in Hidden Layer 1	Neurons in Hidden Layer 2	Accuracy
10	10	**60**
12	8	30
10	5	**60**

56-dimensional input features (MFCCs with SDC features) are processed through 2 hidden layers to give 4-dimensional output vector.

Table 4: Model accuracy for Neural Network with MFCC and SDC features

Neurons in Hidden Layer 1	Neurons in Hidden Layer 2	Accuracy
20	10	50
35	10	70
40	15	**80**

- **Three Hidden Layer Neural Network**
  As the two hidden layered neural network does not provide a reasonable amount of improvement,I have also tried our model with 3 hidden layers with different configurations.Below table shows some results.

Table 5: Model accuracy for Neural Network with MFCC features

Neurons in Hidden Layer 1	Neurons in Hidden Layer 2	Neurons in Hidden Layer 3	Accuracy
12	8	4	30
12	10	8	**60**
12	9	6	**60**

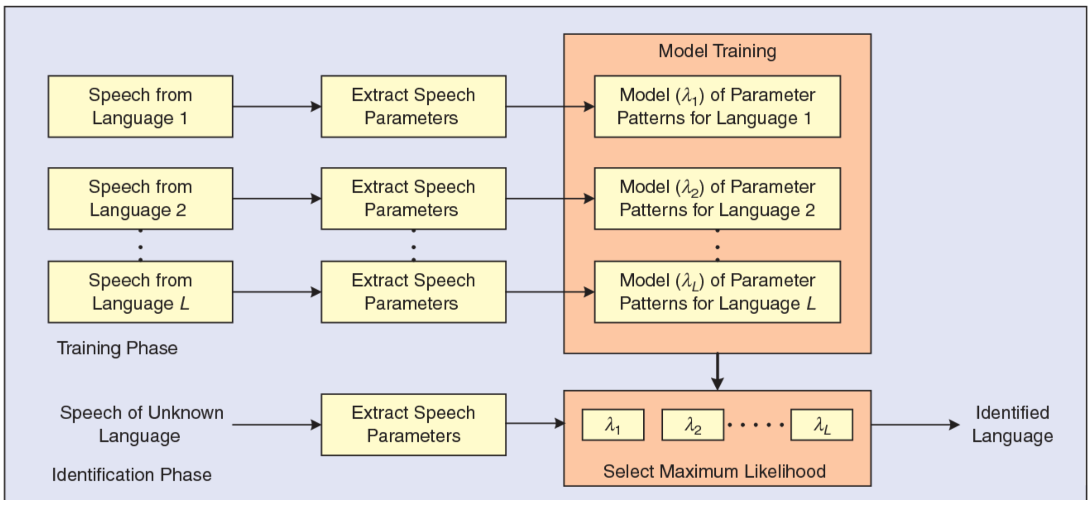
Table 6: Model accuracy for Neural Network with MFCC and SDC features

Neurons in Hidden Layer 1	Neurons in Hidden Layer 2	Neurons in Hidden Layer 3	Accuracy
50	30	10	60
45	30	15	**70**
40	20	10	60
35	20	5	50

This make it clear that our model is not gaining much with increase in number of hidden layers.Thus, it will be total waste of computation power for such improvements.We should look for other methods for improving accuracy of our model.

6 Gaussian Mixture Model for Language Identification

Gaussian Mixture Models are one of the earliest proposed Implicit methods for Language Identification.GMMs simply tokenize the speech signal and language models are derived from the sequence of tokens.GMMs capture language specific information from the extracted features.

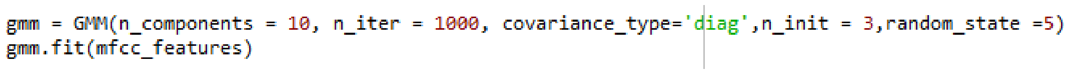


Model Training
Speech from Language 1
Extract Speech Parameters
Model (λ_1) of Parameter Patterns for Language 1
Speech from Language 2
Extract Speech Parameters
Model (λ_2) of Parameter Patterns for Language 2
Speech from Language L
Extract Speech Parameters
Model (λ_L) of Parameter Patterns for Language L
Training Phase
Speech of Unknown Language
Extract Speech Parameters
λ_1
λ_2
λ_L
Select Maximum Likelihood
Identified Language
Identification Phase

Figure 14: GMM based LID system

6.1 GMM parameters

As GMM capture language specific information,thus we train our model for individual languages. We are taking the same speech corpus as above.GMM model is parametrized as:



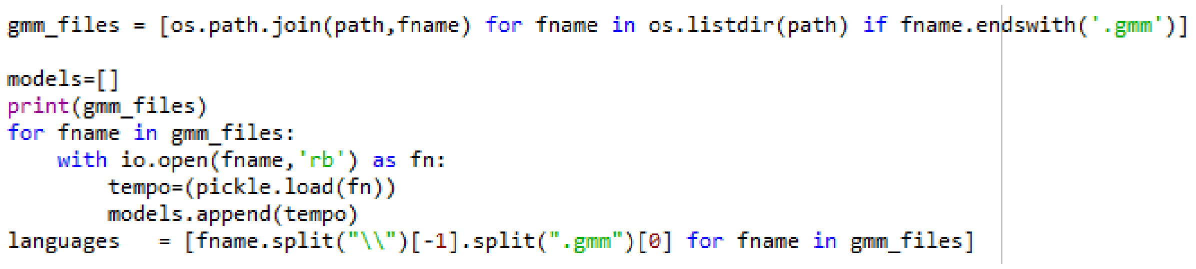
```
gmm = GMM(n_components = 10, n_iter = 1000, covariance_type='diag',n_init = 3,random_state =5)
gmm.fit(mfcc_features)
```

Figure 15: GMM parametrization

n_components - Number of Mixture Components
n_iter - Number of iterations
covariance_type - Types of Covariance parameters.'diag' each component has its own diagonal covariance matrix
n_init - Number of initializations to perform

6.2 Model Training

I have used **MFCCs with delta as our spectral features** for GMM based LID. 40-dimensional feature vector is defined for every sample and language specific model is trained for a particular language.After training our model for individual languages, I have stored our language GMMs into separate files.Thus,4 GMMs are generated corresponding to 4 languages,namely Indian English,Hindi,Bangla and Telugu.

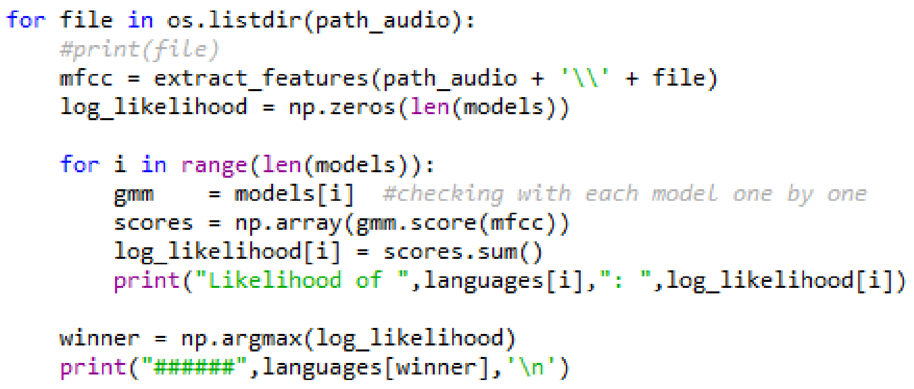


```
gmm_files = [os.path.join(path,fname) for fname in os.listdir(path) if fname.endswith('.gmm')]

models=[]
print(gmm_files)
for fname in gmm_files:
    with io.open(fname,'rb') as fn:
        tempo=(pickle.load(fn))
        models.append(tempo)
languages   = [fname.split("\\")[-1].split(".gmm")[0] for fname in gmm_files]
```

Figure 16: GMM models for each language

6.3 Performance Evaluation

After 4 GMM are generated for 4 language classification,it is evaluated on the testing samples.At the time of model testing,log likelihood scores are calculated for each GMM for given speech sample and maximum of all is identified as the required language.



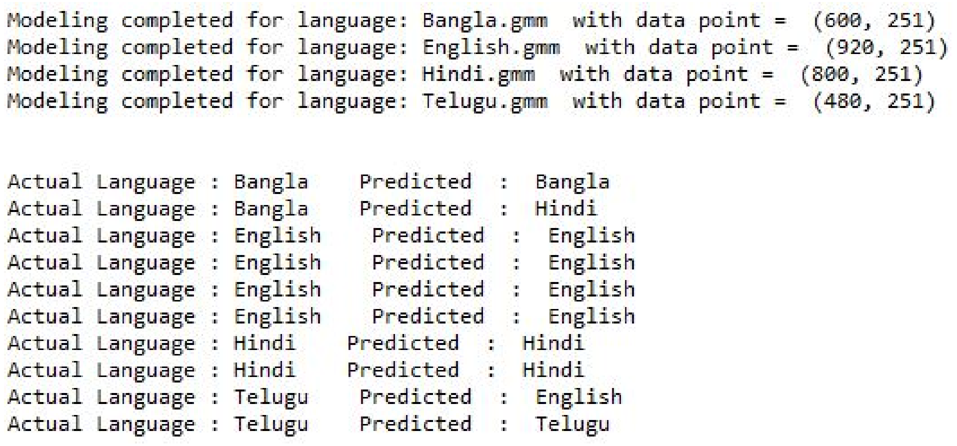
```
for file in os.listdir(path_audio):
    #print(file)
    mfcc = extract_features(path_audio + '\\' + file)
    log_likelihood = np.zeros(len(models))

    for i in range(len(models)):
        gmm    = models[i]  #checking with each model one by one
        scores = np.array(gmm.score(mfcc))
        log_likelihood[i] = scores.sum()
        print("Likelihood of ",languages[i],": ",log_likelihood[i])

    winner = np.argmax(log_likelihood)
    print("######",languages[winner],'\n')
```

Figure 17: GMM models evaluation

Model is tested for different set of features.

- Number of iterations
  Model is not very much affected by number of iterations.1000 iterations is the most optimum number for our model.

- Number of mixture components
  Accuracy of our model also changes with no. of mixture components.For very low number of components and for very high values,it gets disturbed.

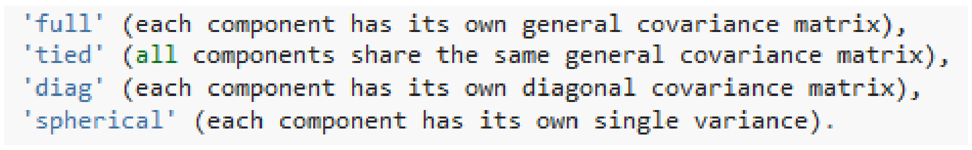


```
Modeling completed for language: Bangla.gmm  with data point =  (600, 251)
Modeling completed for language: English.gmm  with data point =  (920, 251)
Modeling completed for language: Hindi.gmm  with data point =  (800, 251)
Modeling completed for language: Telugu.gmm  with data point =  (480, 251)


Actual Language : Bangla    Predicted  :  Bangla
Actual Language : Bangla    Predicted  :  Hindi
Actual Language : English    Predicted  :  English
Actual Language : English    Predicted  :  English
Actual Language : English    Predicted  :  English
Actual Language : English    Predicted  :  English
Actual Language : Hindi    Predicted  :  Hindi
Actual Language : Hindi    Predicted  :  Hindi
Actual Language : Telugu    Predicted  :  English
Actual Language : Telugu    Predicted  :  Telugu
```

Figure 18: GMM model predictions

- Number of initializations
  Number of initializations has a crucial role in successful mixture estimation using Expectation-Maximization algorithm. Thus,initializations must be made after proper testing over training data.

- Type of Covariance Parameters



```
'full' (each component has its own general covariance matrix),
'tied' (all components share the same general covariance matrix),
'diag' (each component has its own diagonal covariance matrix),
'spherical' (each component has its own single variance).
```

Figure 19: Covariance matrix

I have tested our model for all the covariance parameters.'Full' and 'tied' are not providing any good results.On the other hands,'diag' and 'spherical' have shown encouraging results.I tested our model with these parameters for different number of mixture components. Some of the observations are mentioned below in Table 7.

Table 7: Model Accuracy for different number of components

Number of Mixture Components	'Diag' (Diagonal covariance Matrix)	Spherical(Own single variance)
1	90	80
5	50	40
10	70	80
15	70	60
20	50	60

This clearly shows how the accuracy of model changes with number of mixture components.Thus, mixture components need to be taken carefully after proper validation checking.

7 Conclusion and Future Work Prospects

I have worked extensively on language identification systems,more appropriately Implicit LID systems.Various speech features such as rhythm, melody help us in identifying different languages. Introduction of Neural network has made this task look easy.Different modifications to ANNs and to input MFCCs can help to improve language identification task.

Language Identification system requires a lot of data and improves performance with it.Thus, good quality speech corpus of sufficient size will certainly help our LID system.Many attempts are being carried out for Indian Languages and will surely help.

The accuracy of this model can also be improved by choosing adequate frames from a speech sample which properly show the language characteristic.This will help to differentiate a language from given set of languages.

8 Acknowledgements

I would like to sincerely thank Dr.Vrijendra Singh, HoD, Information Technology, IIIT-Allahabad for guiding me throughout the duration of my work.

I would also like to express my sincere gratitude to Ms. Anjali Gautam, Research Scholar, IIIT-Allahabad for helping me whenever I was stuck with any difficulty during the execution of my project.

9 References

1. Maity, Somnath et al."IITKGP-MLILSC speech database for language identification." 2012 National Conference on Communications (NCC) (2012): 1-5.

2. [Image of Voice Activity Detection(Figure 2). (n.d.). Retrieved from http://practicalcryptography.com/misc
learning/voice-activity-detection-vad-tutorial/]

3. [Image of Forward Propagation in Neural Networks(Figure 9). (n.d.). Retrieved from https://kousikk.wordpress.com/tag/neural-networks/]

4. [Image of Backward Propagation in Neural Networks(Figure 11). (n.d.). Retrieved from https://www.quora.com/Why-use-backpropagation-over-other-learning-algorithm]

5. Ambikairajah, Eliathamby et al."Language Identification: A Tutorial." IEEE Circuits and Systems Magazine 11 (2011): 82-108.